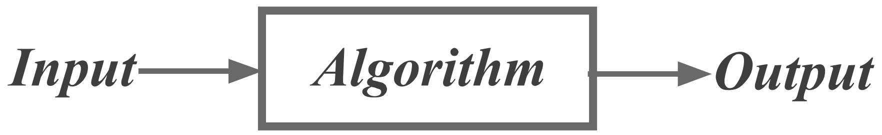
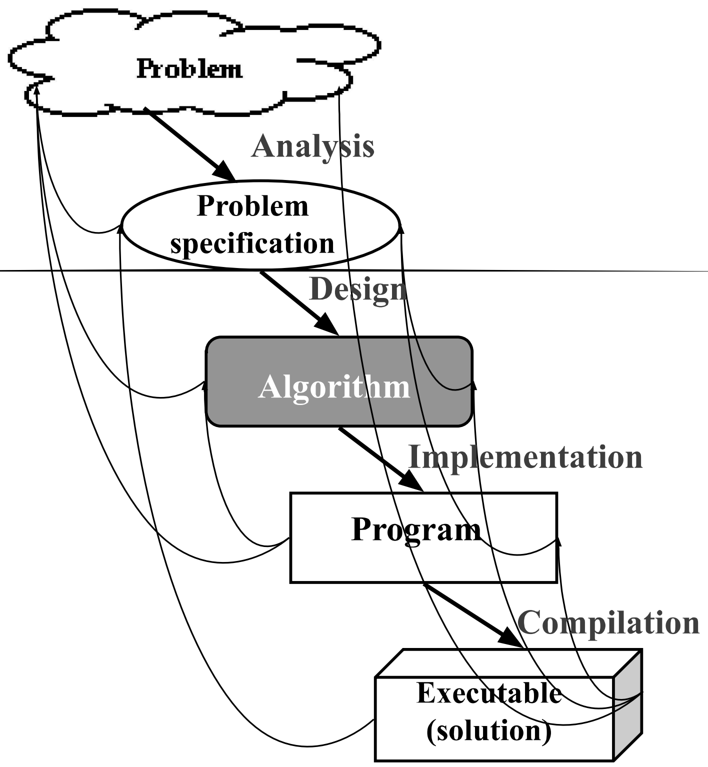
A computer algorithm

- Is a step-by-step method of solving some problem



- Algorithms are the basic ideas behind computer programs.
- An algorithm is the thing which stays the same whether the program is Pascal or C++…etc.

The Problem-solving Process



Pseudocode for algorithms

An algorithm can be written in many ways:

- **English or any language**
- **Pseudocode**

  A way of writing program descriptions that is similar to programming languages but may include English descriptions and does not have a precise syntax

Example: Finding the max of 3 numbers

input=a,b,c. output=x.

```
Max(a, b, c){
    x=a
//if b larger than x, update x
    if(b>x)  x=b
//if c larger than x, update x
        if(c>x)  x=c
Return x
}
```

Example: Finding the max value in an array.

Input: array s. output: max

```
Array_Max1(s[ ], size){
    max=s[1]
    i=2
    while(i<=size){
        if(s[i]>max){
            max=s[i];
            i++
        }
    return max }
```

Example: Finding the max value in an array.

Input: array s. output: max

```
Array_Max2(s[ ], size){
    max=s[1]
    for(i=2 to size){ // i<=size
        if(s[i]>max) max=s[i];
        }
    return max }
}
```

Example: An algorithm for computing the reciprocal of a number

```
Reciprocal(input Num ){
 if (Num is not equal 0)
 {   output 1/Num
 }
 else
 {   output "infinity"
 }
}
```

Example: An algorithm for finding the summation from 1 to a given number

```
procedure Sum1_to_n(num)
{
   count = 1
   sum = 0
   while (count <= num){
      add count to sum
      add 1 to count
   }
}
```

- Note: if **C** is an object of a class containing a function **f**
  - **C.f()** invokes the function **f** on **C**

Two main tasks in the study of algorithms:

- Designing an algorithm to solve a problem
- Analyzing algorithms

In the **analysis of algorithms**, we ask the following questions:

- **Correctness:**
  - does the algorithm solve the problem?
- **Termination:**
  - does the algorithm always stop after a finite number of steps?

- **Time analysis:**
  - how many instructions does the algorithm execute?
- **Space analysis**
  - how many memory does the algorithm need to execute?

In this course, we are concerned primarily with the time analysis.

It is important to be able to estimate the time and space required by algorithms:

- Computers are not infinitely fast and memory is not free, so algorithms must have acceptable time and space requirements
- This allows us to compare algorithms that solve the same problem